**对于E级才能决定写不写or写哪的指令：**

在E级可确定出A3和wdata。

**考虑暂停：**

Tnew=1，clash\_control能够拿到所有正确的A3（clash\_control只需要E，M级的a3来控制暂停），暂停可以不改，如常执行。

**考虑转发：**

对于转发，由于E级Tnew不为0，不转。之后Tnew=0时a3和结果都有，如常转发就可以了。没法转的暂停clash\_control都处理完了。

**总结：**

这种情况无需对CPU进行大改，针对新指令进行控制信号的修改即可。

**对于M级才能决定写不写or写哪的指令：**

**指令类型**

一般有两种，本质上是一类。（1）根据M级结果决定写哪个寄存器；（2）根据M级结果决定写不写。 对于第二种，需要将a3置0，grf\_we同步置0，可以理解为写入0号寄存器。 所以一共就一类：根据M级结果决定写入哪一个寄存器。

对于我的CPU，Tnew=0的条件是已经得出了写入地址和写入数据。那么显然Tgenerator会置Tnew为2。

**考虑暂停：**

两种实现方法：（考场上先做方法1并改乘除周期为3，都为3）

**1）暴力暂停。在E，M级时就暴力暂停。**

实现方法：Clash\_control中新加信号，只要是这条指令，那么就停下：

if( D\_is\_new\_instr==1’b1 || E || M ) stall=1；

优化：只在E级暴力暂停，在M级可以得到所有正确信息，正常使用AT法暂停即可。

**2）只要新指令后面的指令要读 可能要写入的寄存器，并且Tuse<Tnew，那么就暂停。**

对于第二种暂停方法，其控制思想和本来的暂停控制思想几乎一致。在不确定要写入哪一个寄存器的时候，假定都要写，就相当于是认为E\_a3有两个。

可以对这样的指令做特判，实现方法如下：

assign e1\_add = E\_is\_new\_instr & (Tuse1<E\_Tnew) & (D\_a1==E\_a3\_add) & (E\_a3\_add!=5’d0)

assign e2\_add = E\_is\_new\_instr & (Tuse2<E\_Tnew) & (D\_a2==E\_a3\_add) & (E\_a3\_add!=5’d0)

原来的e1，e2不变，那个旧指令要用，新指令也用，所以e1，e2无需改动。

如果指令要写入的寄存器编号一个蕴含在指令中，另一个是固定的，那么很好写，无需加入新接口 E\_a3\_add，直接写31就行，因为E\_is\_new\_instr的存在，其只会在是新指令时起作用。 若指令为决定写不写，实现起来很简单：假设要写，只有一个clash\_control中只有一个a3。

M级可以不使用特判的方法来暂停，使用正常的AT法暂停即可。因为在M级可以得到正常AT法暂停时所需要的确定的a3。

那个E\_a3\_add是在模块内的命名，在datapath中可以用E\_a2之类的连上去。**注**：这样**不**会引入一个新指令本来无需读而被误以为要读而暂停的事件，因为可以置Tuse=3，此时暂停模块一定不出事，转发随便转，反正不用。（不要影响旧指令的行为即可）。 也可以在control中新加一个接口专门用于生成E\_a3\_add

**考虑转发：**

AT法都保证了没有暂停时，当Tnew=0 且 需要转发时转发的结果一定是正确的。

**其它注意：**

对于在M级才能确定A3的情况，显然是需要在control中新加input，在流水寄存器中新加流水。Control中这样写a3的生成assign a3 =( is\_new\_instr && new\_input===1’b1 ) 这样写。那么需要注意D,E两级a3的控制信号中的new\_input是高阻态，要用===来写。assign语句如果遇到了x或者z又没有使用===，整个assign会直接废掉

尽量在control中生成a3的信号，因为这样遵循了高内聚低耦合，而且避免了出现一些小bug的情况，例如，control中的a3不为0，在datapath中改a3为0，但忘了改grf\_we。

**总结要干的：**

如遇第三题为M级才能决定写哪的指令，为了考试好写：

如果E级的特判暂停很好写，就写特判，否则直接暴力暂停。 只考虑对E级进行特判暂停否，修改control以在M级得到正确的a3结果，M级正常AT法暂停。

修改control之后E\_a3要用===，否则会出现E\_a3=xxx的情况。 M级需要生成出一个check信号，用于control中a3的判断，这个信号需要流水。

**考试注意：**

每一道题，考虑：

1.基础实现

2.阻塞暂停，AT

3.转发，有时要转发新东西，例如PC4

4.如果是后面才能得到信号，要用===防止影响前面

CPU的控制模块是由两部分组成的，一部分是control模块，另一部分为clash\_control模块。一有错误两个都要同步改。一加指令两个都要同步加。

如果run too fast 或 too slow，可以手动调整MDU的时间信息。

尽量不在datapath中特判改东西，除非是W级这样的很简单的地方，尽可能遵循高内聚，低耦合的思想。

复杂指令尽量让其融入CPU原路线之中，减少对转发的破坏。如：jap

尝试一下啊，能不能手动修改，利用评测机，如果说错误信息非常靠后，能不能直接强制输出正确信息。

如果是利用 is\_new\_instr 和 assign 来实现多路选择器，那么要把 is\_new\_instr 放到最前面，否则可能无效，因为这条新信号也会产生对应的控制信号。